# Full-Stack Internship Assignment: High-Performance Document Search & Management System

**Objective:** Design, implement, and document a robust, highly efficient, and modern web application that centralizes various data types and provides a high-speed, accurate search interface.

## Project Concept: Document Search & Indexing System

The goal of this project is to provide end-users with a centralized, instantly **searchable repository** of institutional knowledge (Web Links, PDFs, and FAQs). A key design requirement is that the system must be built with **efficiency and scalability** in mind, especially regarding content indexing and search response time.

### Technical Stack (Mandatory)

| Layer | Technology | Usage Focus |
|---|---|---|
| **Backend** | Node.js with **TypeScript Optional** | API development, data processing, robustness. |
| **Database** | PostgreSQL (using only pg library) | Efficient data modeling and built-in search capabilities |
| **API Framework** | Express.js | Creating RESTful services. |
| **Admin UI** | React with Ant Design (AntD) | Rapid prototyping and polished administrative tooling. |
| **Public UI** | React (via Vite) | High-performance, responsive user experience. |

# Backend & Data Architecture

## 1. Database Schema Design (PostgreSQL)

- Design a normalized SQL schema for the application
- **Focus on Performance:** Implement PostgreSQL features (like indexes and constraints) necessary for high-speed lookups and joins.
- **Challenge: Full-Text Indexing:** Implement the best  search mechanism to index the content of all data types.

## 2. API Development (Node.js/TypeScript/Express)

- Develop well-typed, robust RESTful endpoints for all CRUD (Create, Read, Update, Delete) operations for Links, FAQs, and PDFs.
- **Content Processing API:** Create an endpoint to handle PDF file uploads. The server must automatically extract the plain text content from the PDF and Web Page for indexing and store it alongside the file's metadata.

# Admin Dashboard (CRUD Operations)

Using React and AntD, build a secure, administrative interface.

## 3. Data Management Interfaces

- **FAQ Management:** Forms to add, edit, and delete FAQs (Title, Content, Tags).
- **Web Link Management:** Forms to add, edit, and delete links (URL, Title, Description).
- **PDF Management:** Interface for uploading, viewing metadata, and deleting PDF files. Include visual feedback during the upload and processing stages.
- **Efficient Display:** Implement pagination, server-side sorting, and filtering on data tables to handle potentially thousands of records efficiently.

# Public Search Interface

The most critical part of this assignment. The search experience must feel **instantaneous** and **highly accurate**.

## 4. High-Performance Search Implementation

- A single search bar should query content across all indexed data (Links, PDFs, FAQs).
- **Accuracy:** Present the most relevant results first based on the full-text search ranking.

### 5. Efficient Results Display ( Be Creative, No need be exactly like mentioned below )

- Display search results in a clean, responsive layout.
- **Filtering/Faceting:** Implement filters to allow users to narrow results by data type (Link, FAQ, PDF). The filters must be **fast** and update the results instantly without a full page reload.
- **Result Presentation:**
    - **FAQs:** Display the question and a truncated snippet of the answer.
    - **Web Links:** Display the title, a short description, and a clear call-to-action link.
    - **PDFs:** Display the file name and provide a method to view the document. While full inline PDF viewing is a bonus, the priority is an **efficient mechanism** to access the file (e.g., a secure, streamed link).

## Critical Requirement: Focus on Speed, Accuracy, and Efficiency

When developing this assignment, your approach will be judged heavily on the following:

1. **Performance:** How quickly does the search return results? (Aim for sub-200ms response time for typical queries.)
2. **Architectural Efficiency & Research:** Did you critically evaluate and leverage the best tool for the job? While PostgreSQL full-text search is required as a baseline, **be creative and implement best possible solution for "search"**
3. **Code Quality (TypeScript):** Are the backend models and API request/response structures strongly typed and robust?
4. **User Experience (UX):** Is the Public Search Interface responsive

*This assignment is designed to see not only if you can write code, but if you can **think critically** about the architecture needed to deliver a fast and accurate product. Good luck!*